RECEIVED
CENTRAL FAX CENTER
OCT 27 2006

REMARKS

An excess claim fee payment letter is submitted herewith for seven (7) excess claims, although there are only five (5) new claims in the present amendment. Two additional excess claims are included since the documentation in Applicants' representative's file indicates that only twenty (20) claims were alleged in the original patent filing, so that the original filing fee, as charged by the USPTO, may not have accounted for the number of claims actually filed.

Claims 1-27 are all the claims presently pending in the application. Claims 23-27 have been added to address the more general concept of composite blocking, as discussed beginning at line 13 on page 15, and claim 27 is added to address a preliminary conversion of matrix data into the format stored in memory, including the padding of data to fill up blocks, as necessary, and as discussed beginning at line 17 on page 13.

It is noted that composite blocking is the more generic concept of double blocking addressed in claims 4, 11, 17, and 20, which claims are not rejected under the prior art evaluation currently of record. Applicants, therefore, conclude that the Examiner considers these claims to be allowable pending resolution of the issue of non-statutory subject matter.

It is noted that the claim amendments, if any, are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art, narrowing the claims or for any statutory requirements of patentability. Further, Applicant specifically states that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claims 1-22 stand rejected under 35 U.S.C. § 101 as allegedly directed to non-statutory subject matter. Claims 1-3, 6, 7, 9, 10, 12, 14, 15, and 18 stand rejected under 35 U.S.C. § 102(b) as allegedly anticipated by U.S. Patent No. 5,099,447 to Myszewshi. Claims 5, 8, 13, 16, 19, 21, and 22 stand rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Myszewshi.

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

The claimed invention is directed to a method of executing a matrix subroutine. Data for a matrix subroutine call is *stored in a computer memory* in an increment block size that is based on a cache size.

Since standard Fortran and C two-dimensional arrays are stored in memory/processed in either a column or a row of the matrix data, depending on which language is used, there are inefficiencies in retrieving data for sub-blocks of matrix data, as discussed in more detail below.

In contrast, the present invention teaches that the data be stored in memory relative to the sub-block of data that fits into the cache working area and that preferably is stored contiguously in memory, to preclude thrashing of data in cache.

By actually storing in memory the matrix data as contiguous data relative to the block of data and by sizing the block of data relative to the size of the cache, the processing is executed faster than conventional methods, since data thrashing is reduced or eliminated.

Another aspect of the present invention is the recognition that engineering/scientific matrices are typically directed toward square matrices, since such square matrices express linear equations in which a specific solution can be found. In contrast to conventional wisdom, the present invention also teaches that such square matrices can be viewed as rectangular blocks of matrix data that can be brought into cache as a block of contiguous data, again, thereby reducing or eliminating the thrashing that typically occurs when the entire square block is retrieved.

As an example of increased processing efficiency, an application related to the composite blocking of the present invention using contiguous DGEMM operands is Cholesky factorization. Cholesky factorization is an example of a DLAFA. On the IBM Power3 processor the implementation of Cholesky factorization using the present invention achieves 92% of peak performance whereas conventional full format LAPACK DPOTRF achieves 77% of peak performance. Moreover, all programming for the new data structures discussed in the present invention can be accomplished in standard Fortran (or C/C++), through the use of higher dimensional full format arrays. Thus, no new compiler support is necessary to implement the present invention.

II. THE 35 USC §101 REJECTIONS

Claims 1-22 stand rejected under 35 U.S.C. §101 as allegedly directed toward non-statutory subject matter. More specifically, the Examiner considers that claims 1-9 and 19-22 are "*... directed to a computer implemented method of calculation where the inputs are numbers and the results are numbers. Claims 10-14 are although directed to an apparatus broadly encompass a general computer implementing the method. In order for a claimed invention that is directed to such a computer implemented method of calculation, or an apparatus that is no more than a general computer implementing the method of calculation to be statutory, the claimed invention must accomplish a practical application. That is the claimed invention must transform an article or physical object to a different state or thing, or produce a useful, concrete and tangible result ....*"

In response, Applicants submit that a "general computer" inherently provides a concrete and tangible result, as these terms are defined in the Guidelines. Moreover, the usefulness of the present invention is clearly described in even the independent claims, since the present invention increases efficiency of the calculations for linear algebra subroutines by storing the operands <u>in memory</u> in blocks of data based on the dimensions of the cache, preferably as contiguous data in memory and preferably retrieved from memory in increments of line size, such that the data in each contiguous block of data is retrieved as a unit of data.

Moreover, the usefulness of linear algebra processing on computers is clearly discussed at lines 9-12 of page 3 of the disclosure, as having uses throughout scientific and engineering fields and even into games and graphics rendering. The present invention increases the efficiency of the matrix processing by its <u>technique of storing data in memory of the computer as blocks of contiguous matrix data to be retrieved as increments of the block data</u>. The present inventors have recognized that matrix processing is inefficient in Fortran and C, as the matrix blocks, as actually stored in memory for Fortran and C, are typically not contiguous in memory, and the matrix data is likewise normally not stored contiguously in memory, even though the processing of the matrix data will ultimately require matrix data in the order represented by the matrix symbology.

Therefore, Applicants submit that the present invention does indeed provide the "useful, concrete, and tangible result" that is required to be statutory subject matter.

In view of some of the Examiner's wording in the rejection, Applicants suggest that the Examiner might also want to consider what Judge Rich wrote in *State Street*:

"The Supreme Court has identified three categories of subject matter that are unpatentable, namely "laws of nature, natural phenomena, and abstract ideas." Of particular relevance to this case, the Court has held that mathematical algorithms are not patentable subject matter to the extent that they are merely abstract ideas .... In Diehr, the Court explained that certain types of mathematical subject matter, standing alone, represent nothing more than abstract ideas until reproduced to some type of practical application, i.e., "a useful, concrete and tangible result." Alappat. (This has come to be known as the mathematical algorithm exception, especially given the Freeman-Walter-Abele analysis. By keeping in mind that the mathematical algorithm is unpatentable only to the extent that it represents an abstract idea, this confusion can be ameliorated.)

Unpatentable mathematical algorithms are identifiable by showing they are merely abstract ideas constituting disembodied concepts or truths that are not "useful." From a practical standpoint, this means that to be patentable an algorithm must be applied in a "useful" way....

Today, we hold that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces "a useful, concrete and tangible result" – a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades."

As defined by the independent claims, the present invention is addressed to memory management of data and is not at all an attempt to preempt a mathematical algorithm in the abstract. Rather, it addresses the practical problem of thrashing of data and resultant computing inefficiency, as observed by the present inventors, when conventional standard subroutines are processing matrix data. Thus, another way to view the usefulness of the present invention is its result of providing efficiency in the processing of matrices on a computer using such standard matrix subroutines. Contrary to the characterization of the Examiner in paragraph 2 on page 2 of the Office Action, the present, therefore, does indeed provide a "real world result", by it solution to data thrashing during computer execution of standard linear algebra subroutines.

Relative to claims 15-18, Applicants submit that the plain meaning of the claim language of independent claim 15 (e.g., "... medium <u>tangibly embodying</u> a program of machine-readable instructions executable by a digital processing apparatus ....") precludes the Examiner's interpretation. That is, the claim language itself is clearly confined to media which <u>tangibly embody</u> instructions. The Examiner's interpretation simply ignores the plain meaning of the claim language.

In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw this rejection.

III. THE PRIOR ART REJECTIONS

The Examiner alleges that Myszewshi teaches the claimed invention described by claims 1-3, 6, 7, 9, 10, 12, 14, 15, and 18, and renders obvious the invention described by claims 5, 8, 13, 16, 19, 21, and 22. Applicant submits, however, that there are elements of the claimed invention which are neither taught nor suggested by Myszewshi and that the rejection currently of record fails to meet the initial burden of a *prima facie* rejection.

More specifically, the Examiner points to lines 55-60 of column 4. However, Applicants submit that this description merely suggests <u>executing</u> matrix data in units of block data, possibly using block size related to the size of the cache. However, Applicants submit that this description does <u>not</u> satisfy the plain meaning of the independent claims, since there is no suggestion of actually <u>storing the data in memory</u> in these blocks of data in the form of the blocks to be used in the processing. As pointed out above, the execution of standard matrix subroutines will cause thrashing due to inefficiency at the interface of the subroutine.

Therefore, the present invention can be viewed as addressing memory management at a lower level than implied by high level languages such as Fortran and C.

Indeed, the invention of Myszewski is itself <u>constrained to prepare data for the standard DGEMM interface, which uses the Fortran and C data layouts, and has the problems discussed earlier with data thrashing</u>.

The present invention teaches the technique of composite blocking, as articulated in newly added claim 23. The "double blocking" defined in claims 4, 11, 17, and 20 is a specific embodiment of composite blocking. There is <u>no</u> suggestion in Myszewski to convert and <u>store in memory</u> the matrix data of a matrix in rectangular blocks of contiguous data that will each fit into the working area of a cache. The technique of the present invention reduces the data thrashing that typically occurs when matrix data is simply retrieved in its original format as placed by higher level languages such as Fortran and C.

The data thrashing occurs, in the conventional processing, because portions of the matrix data are typically flushed from cache during processing so that additional memory accesses are required as the flushed data becomes needed for current processing and because matrix data is typically not contiguous as actually stored in memory. The present invention addresses this data thrashing by actually storing in memory the matrix data as rectangular contiguous blocks of a size that fits into cache and, if the matrix data is also contiguous and organized on memory line size, reduces the chance that future matrix data will be flushed from cache before it is consumed in the matrix processing. This occurs because the data actually brought into cache from memory will be completely contiguous matrix data (rather than possibly extraneous data in portions of some lines of data) and because the matrix subroutine processing will fit the processing result back into the data currently being consumed, rather than inadvertently flushing out matrix data not yet being processed.

In contrast to the conventional method of dealing with matrix data, the present invention teaches storing the matrix data in memory as actually being rectangular blocks (preferably contiguous to each other) and preferably contiguous data that may even have appropriate padding to fill up lines of memory and/or blocks so that extraneous data is not present.

Moreover, Applicants submit that the rejection currently of record also fails to point to specific lines and columns in Myszewski that satisfy the plain meaning of the language of any of the rejected dependent claims, and respectfully requests that the Examiner provide specific reference to line/columns in the next Office Action. The rejection currently of record simply makes conclusory statements that are not supported in the reference, since the present invention is actually dealing with inefficiencies at a level lower than the orderly process implied by the higher level language implementation of matrix processing, as typically done in Fortran or C.

To better explain the significance of the present invention, the current most commonly used Dense Linear Algebra (DLA) algorithms for serial and SMP processors have a performance inefficiency and hence they give sub-optimal performance. Standard Fortran and C two-dimensional arrays are a main reason for the inefficiency. To correct these performance inefficiencies one can use New Data Structures (NDS), such as the method of the present invention, along with the so-called kernel routines.

The BLAS (Basic Linear Algebra Subroutines) were introduced to make the algorithms of DLA performance-portable. Starting with LINPACK, and progressing to LAPACK, the "Level

1, 2, 3" BLAS were introduced. The suffix i in Level i refers to the number of nested "do loops" required to do the computation of a given BLAS. Almost all of the floating-point operations of DLA algorithms are performed through the use of BLAS calls. If performance were equal to operation count then performance would be truly portable.

However, with today's deep memory hierarchies along with new architectural features, this is no longer the case. To understand the performance inefficiency of LAPACK algorithms, it suffices to discuss the Level 3 BLAS, DGEMM (Double precision GEneral Matrix Matrix). A relationship exists between the Level 3 BLAS and their usage in most of level 3 factorization routines. This relationship introduces performance inefficiency in block based factorization algorithms, and the Level 3 BLAS DGEMM (Double precision GEneral Matrix Matrix) is now discussed to illustrate this fact.

Design principles for producing a high performance Level 3 DGEMM BLAS are given in various references. A key design principle for DGEMM is to partition its matrix operands into submatrices and then call a DGEMM L1 kernel routine multiple times on its submatrix operands. Another key design principle is to change the data format of the submatrix operands so that each call to the L1 kernel can operate at or near the peak Million FLoating point OPerations per Second (MFlops) rate. This format change and subsequent change back to standard data format is a cause of one performance inefficiency in DGEMM. The DGEMM interface definition requires that its matrix operands be stored as Fortran or C two-dimensional arrays.

Any DLA factorization algorithm (DLAFA) of a matrix A calls DGEMM multiple times with all its operands being submatrices of A. For each call, data copy will be done. Therefore this unit cost gets multiplied by this number of calls. However, this overall cost can be eliminated by using a new data structure, such as taught by the present invention, to create a substitute for DGEMM; e.g. its analogous L1 kernel routine, which does not require the aforementioned data copy.

The Myzewski patent describes standard cache blocking. The ideas behind standard cache blocking were first discovered and reduced to practice during 1984 and 1985 by the IBM Engineering and Scientific Subroutine Library (ESSL) programming team. In February 1986 IBM released the ESSL programming library to the public. The designers of ESSL recognized the need for level 3 BLAS and helped to propose as an Industry Standard the Level 3 BLAS. ESSL in had a version of DGEMM even before the standard was adopted.

Myzewski does not mention any format changes to DGEMM's operands. It is a well known fact that certain DGEMM operands will thrash any L1 cache. By using the data structure of the present invention, this thrashing problem is at least reduced, if not eliminated.

The present invention is directed at improving the performance of DLAFA. In the high performance DLA community, BLAS 3 were invented to make DLAFA run faster. To avoid the performance flaw described above, one can use the new data structure (e.g., memory storage technique) described in the present invention. Composite blocking, the subject of the present invention, is a means to increase DLAFA performance when using Square Blocks (SB). Note that a SB is a fundamental building block of the data structure. An innovation in the present invention is to realize that a SB can also be viewed as rectangular block. Also, rectangular blocks are preferred by DGEMM kernels as its function C = C - AB has an asymmetry.

One key feature of the present invention is that input operands of all DGEMM calls be stored as <u>contiguous</u> data <u>in memory</u>. A major use of the present invention is for producing high performance Dense Linear Algebra Factorization Algorithms (DLAFA), since the full format data structures of Dense Linear Algebra (DLA) hurt the performance of its factorization algorithms. Full format rectangular matrices are the input and output of the level 3 BLAS.

It follows, therefore, that the LAPACK and Level 3 BLAS approach has a basic performance flaw. By using composite blocking described in the present invention, this flaw can be removed.

In contrast, Myszewski was directed towards standard cache blocking, since the inputs to BLAS3 DGEMM are standard full format arrays.

As mentioned earlier, measurable improvement in efficiency has been measured by using the present invention for Cholesky factorization as an example of a DLAFA, wherein, on the IBM Power3 processor Cholesky factorization achieves 92% of peak performance, whereas conventional full format LAPACK DPOTRF achieves 77% of peak performance.

Therefore, Applicants submit that there are elements of the claimed invention that are not taught or suggest by Myszewski, and the Examiner is respectfully requested to withdraw these rejections.

RECEIVED
CENTRAL FAX CENTER
OCT 27 2006

Serial No. 10/671,887
Docket No. YOR920030010US1 (YOR.424)

IV. FORMAL MATTERS AND CONCLUSION

The disclosure has been amended to update the information relative to the co-pending applications.

In view of the foregoing, Applicant submits that claims 1-27, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: 10/27/06

[illegible]

Frederick E. Cooperrider
Registration No. 36,769

McGinn Intellectual Property Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254